Definition of an FHIR-based multiprotocol IoT home gateway to support the dynamic plug of new devices within instrumented environments

Paolo Zampognaro¹ · Giovanni Paragliola² · Vincenzo Falanga¹

Received: 6 September 2021 / Accepted: 4 November 2021 / Published online: 7 December 2021
© The Author(s) 2021

Abstract

Internet of Things (IoT) technologies have become a milestone advancement in the digital healthcare domain, since the number of IoT medical devices is grown exponentially, and it is now anticipated that by 2020, there will be over 161 million of them connected worldwide. Therefore, in an era of continuous growth, IoT healthcare faces various challenges, such as the collection over multiple protocols (e.g. Bluetooth, MQTT, CoAP, ZigBEE, etc.) the interpretation, as well as the harmonization of the data format that derive from the existing huge amounts of heterogeneous IoT medical devices. In this respect, this study aims at proposing an advanced Home Gateway architecture that offers a unique data collection module, supporting direct data acquisition over multiple protocols (i.e. BLE, MQTT) and indirect data retrieval from cloud health services (i.e. GoogleFit). Moreover, the solution propose a mechanism to automatically convert the original data format, carried over BLE, in HL7 FHIR by exploiting device capabilities semantic annotation implemented by means of FHIR resource as well. The adoption of such annotation enables the dynamic plug of new sensors within the instrumented environment without the need to stop and adapt the gateway. This simplifies the dynamic devices landscape customization requested by the several telemedicine applications contexts (e.g. CVD, Diabetes) and demonstrate, for the first time, a concrete example of using the FHIR standard not only (as usual) for health resources representation and storage but also as instrument to enable seamless integration of IoT devices. The proposed solution also relies on mobile phone technology which is widely adopted aiming at reducing any obstacle for a larger adoption.

Keywords IoT · Sensors · Home gateway · BLE · MQTT · HL7 · FHIR

1 Introduction

Internet of Things (IoT) technologies have become a milestone advancement in the digital healthcare domain, since the number of IoT medical devices is grown exponentially, and it is now anticipated that by 2020, there will be over 161 million of them connected worldwide. As properly analysed in [1] Internet of Things (IoT) technologies are increasing

rapidly, having become a milestone advancement in the digital healthcare domain.

This huge expansion of the IoT medical devices market is due to the evolution of high-speed networking technologies, and the increasing adoption of wearable devices, smartphones, and other mobile platforms in healthcare. Therefore, nowadays there exists a large amount of IoT medical devices, which is gradually increasing resulting into a myriad of heterogeneous devices that are connected to the healthcare IoT world. This represents a barrier for health organisation to concretely prepare and implement telemedicine solutions that seamlessly integrate with their HIS (Health Information System). The barriers spread over several dimensions that should be properly addressed.

The *first dimension* relates to the fact that sensors can have different transmission modalities. Some of them send their data over a protocol (e.g. BLE, MQTT) without applying any additional data transformation so that the first integration step

✉ Giovanni Paragliola
giovanni.paragliola@icar.cnr.it
Paolo Zampognaro
paolo.zampognaro@eng.it
Vincenzo Falanga
vincenzo.falanga@eng.it

¹ Engineering Ingegneria Informatica S.P.A. ICT4Health Lab, Rome, Italy

² ICAR, CNR, Naples, Italy

consists in implementing, within a gateway, a middleware able to read data over such protocols. However, in a growing number of cases (often for business reasons) sensors are strictly coupled with an app (mobile app usually) which is the only one able to extract the data over protocols. It sends data to a remote cloud server (e.g. Google Fit) so that the retrieval of data is only allowed in a second moment by interacting with REST service API offered by such cloud server. Without tackling such “data access” heterogeneity modalities an IoT platform risks to be not flexible enough.

The *second dimension* is associated to the gap between the original data format, once they are acquired, and the health standard format adopted by the the HIS. Without such kind of adaptation, a huge amount of data risk to be not really exploitable by an health institution neither for patient care or for medical research. Finally, it must be taken into account, as *third dimension*, the degree of intrusiveness of a solution (also from an economical point of view) that can have an high impact on its real adoption by hospitals and by the final users (i.e. patients).

Starting from this context, to address these issues, the contribution of this paper is the definition of a smartphone-based IoT Home Gateway architecture implemented in the project AMICO and based on HL7 FHIR standard. The main aspects of the architecture are the interoperability and the extension simplicity which allow transparent interoperability in a way that devices can use its own protocol (currently BLE, MQTT) to have a heterogeneous communication.

The rest of the paper is organized as follows. In Sect. 2, we review recent literature of IoT Gateway solutions while in Sect. 3, a general overview of the AMICO project is provided. Sections 3.1 and 4 present the general picture of the IoT Home Gateway (HG) architecture while details about the FHIR based semantic annotation process is described in Sect. 5. Section 6 outline a mechanism supporting the dynamic plug of third party BLE reading modules that further improve the proposed approach. Finally, a brief summary of the main outcomes is outlined in Sect. 7.

2 Related work

We have analysed several IoT gateway solutions to evaluate the coverage degree against the dimensions described before. Figure 1 summarizes the state of the work.

In [1] is proposed a solution able to identify the similarities that exist between the different attributes of the HL7 FHIR resources and the attributes of the healthcare related datasets, and finally transform them into the HL7 FHIR format. Although the data interoperability is a common objective (*second dimension*) in our approach we exploit directly a FHIR resource (i.e. DeviceDefinition) to link protocol specific structures appointed for data transportation (e.g.

characteristic fo BLE) to the Device Definition capability attribute exploiting well known (health) coding systems (e.g. SNOMED, LOINC).

Other authors [2] propose an open-source multiprotocol gateway which supports two communication protocols, namely, Bluetooth Low Energy (BLE) and LoRa. This is inline with our objective (*first dimension*) of eliminating the need of using multiple hardware components for each type of protocol. However, it does not cover data access by cloud services and is a general purpose solution not tackling the data format adaptation toward health standards (*second dimension*).

The work carried out in [3] is more aligned with our proposal since they present a solution to cover protocols and data access heterogeneity (*first dimension*) and they also release a smartphone-centric mobile gateway following the paradigm “my world in my packet” (*third dimension*). However, no specific step ahead is accomplished toward a seamless integration with HIS from a data format adaptation point of view (*second dimension*).

In [4], the authors developed a software architecture to bolster a total smartphone-centric gateway application and executed a real testbed to measure the exhibitions of the proposed solution in terms of energy utilization and assets occupancy in several stack conditions with the point of drawing noteworthy discourses on the real possibility of such integrated communication design.

In [5], the authors proposed an online cloud-based framework to estimate battery lifetime of a smartphone device that consists of two layers; one running on smartphone side, whereas the other residing on the cloud server offering two modes for the energy estimation of smartphone applications to ensure its high applicability.

da Cruz et al. [6] proposed MiddleBridge, an IoT application layer gateway that converts MQTT, CoAP, Websockets, and DDS messages into HTTP. MiddleBridge can be deployed on any computer that runs a Java virtual machine and allows IoT gadgets to communicate with a Middleware seamlessly.

In [7], the authors proposed a security framework for the IEEE 802.15.4 standard, namely LICITUS. The authors performed experiments that have been also carried out to demonstrate the real effectiveness of the conceived solution and to scavenge its possible down sides. It has been shown that all the secured operations provided by the proposed framework require a not negligible computational effort and generate an increment of the energy consumption.

Kumar and Dezfouli [8] showed how utilizing QUIC [9]) in IoT scenarios. The authors integrated MQTT with QUIC in order to justify the potential benefits of QUIC compared to TCP/TLS and UDP/DTLS in IoT scenarios and presented its integration with MQTT protocol.

Table 1 BLE vs Bluetooth

Technical specification	BLE	Bluetooth
Radio frequency	2.4 GHz	2.4 GHz
Distance	10–100 m	10–100 m
Symbol rate	1–3 Mbps	1 Mbps
Application throughput	0.7–2.1 Mbps	305 Kbps
Nodes/active slaves	7	Unlimited
Security	56 to 128bit	128bit AES
Robustness	FHSS	FHSS
Latency (from not connected to send data)	100 + ms	< 6ms
Government regulation	Worldwide	Worldwide
Certification body	Bluetooth SIG	Bluetooth SIG
Voice capable	Yes	No
Network topology	Point-to-point, scatternet	Point-to-point, star
Power consumption	1 (reference value)	0.01–0.5 (use case dependent)
Service discover	Yes	Yes
Profile concept	Yes	Yes
Primary use cases	Mobile phones, headsets, stereo, audio , automotive, PCs etc.	Mobile phones, gaming, PCs, sports, medical, automotive, industrial, automation, home electronics etc.

introduces several standards to manage the **physical** and **MAC** layers of the wireless technologies currently adopted. Really, the standards which are at the base of WPAN are associated to IEEE projects which falls down in the category: 802.15. Among these, ones those related to a mid (or low) transmission frequency (Low Rate)—that characterize also the AMICO context—are mainly the following:

- **IEEE 802.15.1 BLE** It standardizes the physical and MAC layer of both the Bluetooth standard protocol and the BLE (Bluetooth Low Energy) by managing in a different way the network and application layers of the transmission stack. BLE had a wide diffusion mainly due to the low energy consumption respect to the traditional Bluetooth.

This is really appreciated in all that applications that needs to exchange limited amount of data over a long period. In fact, such devices are able to properly run for some years exploiting small batteries. Furthermore, it has been conceived to remain constantly in sleep mode until a connection is triggered. In this respect, the time interval for the pairing step is of a few ms against the 100 ms of the traditional Bluetooth. Such reduction of the pairing time interval is due to an increase of the data transmission speed that is of 1Mb/s for the BLE (see table I). By continuing the comparison between the two technologies (see table I), it is clear that also the application throughput is around of 100–250 Kbps that is one order of magnitude lower respect to the Bluetooth.

This comparison demonstrated the main advantages of BLE especially when devices access and low energy consumption must be guaranteed. This made this standard one of the main protocol adopted in IoT scenarios also in health domain. In fact, the most part of (off the shelf) health devices nowadays (body temperature sensors, blood pressure sensors, body weight sensor are only some examples) adopt such standard. Also, in AMICO project, the new designed sensors exploit the BLE protocol e.g. the environmental parameters sensors (from ST Microelectronics), the ultrasound sensor (from University of Firenze).

- **IEEE 802.15.4 protocols** Indeed the 802.15.4 standardizes the MAC and physical layers for several IoT protocols. Each one of such protocols standardizes the higher layers (till to the application one—see Fig. 3). Our analysis focused mainly on ZigBee and 6LoWPAN. The first one for the great diffusion and the achieved especially in the domotic applications.

The second because it enables the same higher level protocols expected by the web technology. ZigBee has the disadvantage that devices can interact among them exclusively if they are all ZigBee devices since the network and application layers are both defined by this proprietary specification. Moreover, no device, involved in the project, was a ZigBee device. For this reason, the attention focused mainly on 6LoWPAN. The 6LoWPAN adopt IPv6 protocol for communication (transport level). This way devices adopting IP technology can easily connect

Fig. 3 Protocols over 802.15.4

to IP networks without the need to rely on dedicated gateway or proxy.

However, the protocols exploitable on top (i.e. application level) are several and we focused in the end to the MQTT being its great diffusion especially among IoT developers.

From the analysis carried out the following requirements arise:

- **Req 1.** The platform must be able to collect data over IEEE 802.15.1 (BLE—Bluetooth Low Energy) standard managing the connection on the whole ISO-OSI stack till to the application level (i.e. GATT).
- **Req 2.** The platform must be able to collect data over IEEE 802.15.4 supporting MQTT as main application protocol.

Data access heterogeneity: Nowadays, an increasing number of sensors are enabled to transmit data in a *direct* way in some cases or *indirectly* in other cases e.g. some devices send the data to remote servers and Cloud REST API make possible, to an external system, the data retrieval in a second moment. For this reason, an IoT platform should support data acquisition over multiple protocols both in a *direct* way and to automatically interact with cloud servers making data available for retrieval in a completely transparent way to the server applications. In this respect, our attention focused on two of the most adopted solutions: Google FIT [16] and Samsung Health [17].

We compared such cloud-based systems for the management of health data using different specific metrics such as Supported (mobile) operating systems, integration constraints/limitations, support on custom health data types definition, evaluation of Cloud REST API and related integration modalities and, finally, the evaluation of mechanisms in place to manage the sensitive data access. From the analysis and comparison, we choose for the Google FIT cloud solution so as to it is more flexible than Samsung health system resulting in full-fledged in a wider range of technologies.

Moving on to the analysis of the integration possibilities of smart object devices (e.g. smart band), both the solutions require a development effort to allow the mobile app (generally always provided by the device manufacturer) to communicate with a specific middleware layer. Then, in terms of development effort, we did not catch a big difference. It is also worth to mention that Samsung tries to speed up and simplify the integration of new smart objects offering an “SDK for device” [18] which allows to develop a smart object firmware enhancing the BLE GATT profile. The smart object developed with such SDK will be directly accessible by the Samsung Health mobile app so that the manufacturer does not need to develop a dedicated mobile app. Although this is a valuable intention, it is clear as well that this approach could reduce the effective interoperability since these smart object will be not accessible anymore by GATT standard receiver (GATT service and characteristic UUIDs are different from the standard ones) [19].

With respect to the remote server access by REST API, although it is guaranteed by both the solutions, Samsung health only provides access through its Android SDK. This means that the development of final applications in a different technological environment (e.g. web applications) is not allowed. About security aspects Google technology in our understanding seems to guarantee a deeper control of sensitive data access. This means a developer needs to apply for OAuth API [20] verification for sensitive scopes on top of the standard authorization permissions for Android and REST to read and write to these data types [21] (<https://support.google.com/cloud/answer/9110914>). The other dimensions (data types and support for custom data types) did not point out relevant difference.

The results of the analysis have been synthetically summarized in the Table 2. From the analysis carried out, the following platform requirements arise:

- **Req 3.** The platform must be able to collect data from devices compliant with GoogleFit cloud platform. This means the capability to interact with GoogleFit remote rest API.

Data collection and health data format compliance: A lot of commercial devices are available in the market offering the capability to collect data from home sensors such as Zigbee 3.0, Zwave, BLE, Z-Wave, EnOcean. In a similar way, the SWYCS is a universal and vendor neutral infrastructure that connects wireless equipment and systems based on open standards. It is a multi-protocol and interoperable IoT gateway connecting wireless ZigBee, Z-Wave, EnOcean and BLE equipment and systems.

The mentioned devices are only some examples available in the market offering similar capabilities. However, these solutions require a dedicated hardware to ship from

Table 2 Cloud health data management systems comparison

METRIC	Google FIT	Samsung Health
Supported (mobile)operating systems	Android, iOS	Android
Smartwatch/wearable direct integration	YES (Wear OS, Tizen)	YES (Tizen)
integration constraints/limitations,	All device whose applications integrate FIT services	All device who comply with SDK and those integrated directly by Samsung
Supported data types	51 grouped in 8 category [16]	48 grouped in 7 category [17]
Support custom data types	YES	YES
Cloud REST API and Integration	YES (direct access and Android SDK)	YES (only by Android SDK)
Sensitive data management	Regulation by google's security processes and user acceptance	Regulation by user acceptance

the market and to install in the home environment requiring a dedicated budget and appropriate technical skills for their configuration. Furthermore, they have been conceived to made sensors data available to external system without any data adaptation. This represent a further obstacle for a rapid integration of sensors data within the Hospital Information Systems (HIS) that are generally compliant with consolidated standards.

Effective use of informatics systems and tools by providers and their patients is key to improving the quality, safety and cost of healthcare. With health records now digital, no effective means has existed for sharing them with patients, among the multiple providers who may care for them and for important secondary uses such as public/population health and research. This problem is a topic still tackled in several congressional discussions and has been also addressed by the 21st Century Cures Act of 2016 that mandated that electronic health record (EHR) systems offer a patient-facing API. HL7's Fast Healthcare Interoperability Resources (FHIR) HL7 standard is that API and it is already being used in several real contexts.

FHIR is based on web technologies and is thus a far more facile, easy to implement approach that is rapidly gaining acceptance. It is also the basis for a 'universal health app platform' that literally has the potential to foster innovation around the data in patient records similar to the app ecosystems smartphones created around the data they store. An increasing number of health organisations are adapting their systems to be compliant to the HL7 FHIR standard.

From the analysis carried out, the following platform requirements arise:

- **Req 4.** The platform must be able to collect data in an unobtrusive way and simplifying the rapid setup of the instrumented environment by the health organization.
- **Req 5.** The platform must be able to adapt the native device data format to the HL7 FHIR standard so to guarantee a seamless integration into health organization information systems.

Table 3 Android—iOS global market share (Sept 2020) [22]

Country	Android (%)	iOS (%)
Europe Italy	84.6	15.3
Europe Spain	90.7	9.3
Europe Germany	80.9	18.9
Asia Cina	87.4	12.3
Asia Japan	62.7	36.9
Oceania Australia	59.3	40.7
South America Argentina	97.8	2.2
North America USA	57.7	42.2
North America Canada	48.75	50.89

4 FHIR-based IoT gateway architecture

The analysis described in the previous section convinced us to design and implement the AMICO home gateway by exploiting a mobile phone technology without forcing patient or health organization to buy dedicated hardware and learn specific set up procedures (see Req 4). In this respect, our attention focused on Android-based mobile phone considering the greater diffusion worldwide (see Table 3).

This choice was also justified by the availability of an android tablet provided along with the Pepper robot. In Fig. 4, the overarching architecture is represented and, following, the main components and their interaction are briefly described. The next sections, instead, are dedicated to present the details about the specific components enabling (1) the automatic harmonization of data format toward HL7 FHIR, (2) the run time plug of new BLE devices. To simplify the reading of the architecture, the different data flows have been highlighted: the dotted white arrows refer the input data flows from sensors while the blue arrows are related to other kind of data flows (i.e. device configuration and authorization steps). By analysing the figure from the top right side and following the arrows, the ProtocolManager component is appointed to collect data coming from BLE and MQTT (see Req1,

Fig. 4 AMICO sensors home gateway-architecture

Req2) devices and from GoogleFit cloud service (see Req 3) and send them to the SensorsGatewayService. The SensorsGatewayService is the background Android service that forwards the received raw data to the FhirAdapter component that implements the logic to automatically transform the data into FHIR Observations (see Req 5). The SensorsGatewayService, finally, exploits the ObservationUtils component to send the generated observations to the ClinicalWebApp that allow the clinical teams the access to the monitoring data.

Moving to the bottom side, it is possible to recognize the user interfaces (Android UI) exploited by a DeviceManager to accomplish some configuration activities and functioning check. For instance, the two UI SensorsBLE Fragment and SensorsMQTT Fragment retrieve, through the DeviceUtils component, the device information (i.e. the FHIR Device and DeviceDefinition resources) available at Health Organisation FHIR server. They also get from the SensorsGatewayService the IDs of the sensors dynamically discovered in the surrounding environment. This way such interfaces are able to inform the DeviceManager about several details as: the sensors currently running in the environment, if they are owned by the Health Organisation or by the patient, if they are currently associated to the patient CarePlan or not etc.

Moreover, once selected a device the same interface allow to set some configuration information as, for instance, the position of the sensor on the body of the patient (i.e. the body part) or the specific environmental location (e.g. kitchen, dinner, bathroom, etc.). At the end, such information are pushed back by the DeviceUtils so that they will be always available to the clinicians.

It is also worth to mention that if the device selected for the configuration is not part of the Health Organisation fleet but it is a personal device of the patient then, through the same interface, it is possible to submit an approval request to the Health Organisation team. The gateway (always running as an Android background service) periodically checks the approval status change and when the device is enabled by clinicians, then the gateway starts collecting data also from the patient personal device.

By means of the SensorsGFITFragment, instead, it is possible to perform the authorization steps to enable the mobile app to retrieve the patient data from the GoogleFit cloud service. Finally, by means of the MonitoringFragment interface the FHIR observations produced starting from the sensors raw data are showed in real-time offering, such a way, a useful instrument to check the correct operational status of the gateway.

Fig. 5 BLE-FHIR FHIRAdapter

5 BLE-FHIR conceptual mapping: FHIRAdapter

In this section will be presented the analysis carried out to find a *mapping* between a BLE device data structures and the FHIR resources appointed to capture the information related to a device and its description.

Such a mapping will represent the theoretical model implemented by the *FhirAdapter* to automatically transform BLE raw data into FHIR Observations.

The main FHIR resource involved in the mapping is the *DeviceDefinition*. In fact, each FHIR *Device* resource contains the attribute *definition* referring a *DeviceDefinition*. The *DeviceDefinition* is a resource able to describe the generic structure of a device (for instance a pulseoximeter), by providing some properties that characterize it and specific functional properties (i.e. *capabilities*).

So in the first phase of the research activity, we focused on a comparative analysis between the structure of the (BLE) GATT profile and the attributes of the *DeviceDefinition* trying to identify the direct correspondences and the needed extensions. Since the two data structures resulted quite soon really different, we adopted a conceptual mapping tool (i.e. MindMaster) to carry on the analysis in a simple way. In Fig. 6, the first mapping from a BLE service to FHIR is showed. To be as concrete as possible we will rely on the Heart Rate Service to show the whole mapping process. The FHIR *DeviceDefinition* does not contain, natively, any attribute directly exploitable to represent a BLE service. For this reason, such information has been captured by means of an ad hoc FHIR *Extension* (see “Extension [Service]” in Fig. 6), composed, in turn, of further extensions.

In detail, the following BLE service informative content has been captured by FHIR Extension:

- **Uniform Type Identifier:** Unique string that identifies the BLE service and that adopts a domain naming convention—e.g. org.bluetooth.service.hearth_rate.
- **Name:** The name of the BLE service—e.g. Heart Rate.
- **Assigned number:** The unique identifier of the BLE service - e.g. 0x180D.

- **Characteristic:** It represents a ‘container’ holding the information about the allowed operation.

Really, the information extracted from the BLE service and captured in a FHIR Definition are only used for checking purpose but are not strictly mandatory to enable the transformation process from BLE raw data to FHIR observations. The core mapping, instead, is that one related to the BLE characteristics. As shown in Fig. 8, each BLE characteristic information is mapped to a *DeviceDefinition capability*, either exploiting native attributes or introducing the needed extensions to the *capability* inner object.

Every modelling lack in the *DeviceDefinition capability* object was resolved using the proposed extensions, properly conceived and modeled to capture the different nature that the BLE data can assume. Since the BLE characteristic *Assigned Number* is a unique hex identifier, it can be represented by string values identified, in the FHIR domain, by a unique extension url that, in our implementation, is “<http://bluetooth.org/assigned-number>” (see Extension [Assigned Number] in Fig. 8).

The (field) *Unit* is one of the most important information captured by a *DeviceDefinition* related to a BLE Characteristic and allows to specify the BLE characteristic (measurement) unit of the *value field*. Really, this extension could also be used to hold a desired unit of measure, distinct from that one carried by the BLE characteristic, that will be used by the Home Gateway to perform the needed adaptation during the sensor data acquisition process. This information is modelled using a *CodeableConcept* that is a data type able to contain different kind of information depending from a specific code associated to a coding system. In this case, we adopted a well known coding system for measurement units (i.e. UCUM [23]) while the extension URL was “<http://amico-default-measure.org>” (see Extension [Unit]).

Flags is the section dedicated at all the options of the BLE Characteristic. For example, if a device has more than one field name providing the same kind of measure but with different accuracy degree, it is possible to state, by means of the bit flags section, which one is expected to be received. We have modelled this informative content about flags by means of an extension (which URL is “<http://bluetooth.org/characteristic/flags>”) composed, in turn, of heterogeneous extensions in order to properly catch the bit flags nature. For instance, if the BLE bit flag is a boolean value, the data type of the extension value will be a Boolean data type and so on (see Extension [Flags] in Fig. 8).

A *Field Name* specifies the name of the BLE characteristic *value field*. In general, it is possible that more than one *value field* are carried by a characteristic but not all of them necessarily transport measurement data. Furthermore, not all *value field* must be present in a BLE transmission. As said

Fig. 6 BLE-FHIR service

Fig. 7 BLEPM ecosystem interaction

before in this section, the bit flags can inform the receiver about which *value field* are actually in use or which accuracy they support.

More in general, every *value field* can rely on one bit flag. In our solution, this information is carried by an extension (see Extension [Field Name] in Fig. 8) identified by the URL “<http://bluetooth.org/characteristic/field-name>” and is modeled by a string. Such a string informs about the name of one or more *value field*—in the second case, the string is a chain of the names of each *value field*, each name being separated by a semicolon.

An important assumption (the DeviceManager has to take care during the DeviceDefinition creation) is that in

the case of multiple *value field* involvement, then these fields must have the same health meaning [e.g. HeartRate (8 bit), HeartRate (16 bit)]. If, instead, one characteristic transports multiple heterogeneous health measures—that is allowed by BLE although discouraged—(e.g. HeartRate and GlucoseLevel), the DeviceManager has to use two distinct DeviceDefinition capabilities and place the names of the two *value field*, separately, into the filed name extensions of the two capabilities.

This is due to the fact that our approach aims at capturing the semantic of each sensor health measures by means of the *type* of a single DeviceDefinition capability which transports the semantic of a device “measurement capability” and it is a FHIR *CodeableConcept*. Through such codeable concept we rely on well known standard health vocabularies (e.g. SNOMED, LOINC) by which it is possible to express any concept without ambiguity and, most important, in a machine understandable modality. In the Table 4, a resume of the mapping just discussed.

All these information, encoded in DeviceDefinition’s capability, must be preliminary provided by the DeviceManager through a DeviceDefinition *editing step* accomplished by means of a dedicated area of the AMCO ClinicalWebApp (see Fig. 4). At the end of this step, the new DeviceDefinition is persisted in a FHIR server and its informative content will allow our algorithm, running in the Home Gateway, to be able to produce a correct transformation from the data received over BLE, to a FHIR Observation resource that will be finally stored on the FHIR Server.

Fig. 8 BLE-FHIR characteristic **Table 4** BLE characteristic—DeviceDefinition capability

FHIR DeviceDefinition/capability	BLE characteristic
Extension[Assigned number]	The unique identifier (i.e. assigned number) of the i-th characteristic (e.g. 0x2A37)
Extension[Unit]	The measurement unit associated to the transported data value
Extension[Flags]	The parameters (i.e. bit flags) which allow to specify information about how to read the data
Extension[Filed name]	The name of the value field(s) containing the health measurements data

[1]The capability *type* attribute is appointed to hold the meaning of the health measure. The capability *description* attribute provides further details about the capability

More in details, from the image below (see Fig. 5) it is possible to identify three distinct blocks, each one representing a task of the transformation algorithm, relying (1) on a piece of the DeviceDefinition associated to the sensor and (2) on a piece of data received from the sensor itself. The piece of information provided by the DeviceDefinition is represented by the triangle icon, while the circle icon represents the received data extracted from the BLE channel and named from now on **BLE Measurement**.

The first block represents the ability of our algorithm to search and assign the correct code to the received measure. The code must identify uniquely the meaning of the measure (i.e. its semantic) without any ambiguity. It can be associated to a proprietary coding system or coming from existing health vocabularies as, for instance, SNOMED, LOINC, etc. In our solution, the code has been included in the *type* property of the DeviceDefinition *capability* property during the editing.

To retrieve the code corresponding at the received measure, we compare the characteristic assigned number, extracted

from the BLE Measurement, with the correspondent one stored in the assigned number extension held in the capability property of the DeviceDefinition resource. Once a match is found, then the code, retrieved from the capability type property, is used to set the *code* property of the generated FHIR observation that will be sent to the server. In this way, we can always know what type of measure was read from BLE sensor.

The second block represents the ability of our algorithm to search and assign the correct unit of measure code to the received measure. This code can be a proprietary code, but we suggest to use a standard code defined by international system that can be widely understood. The default unit of measure, expressed by the DeviceDefinition creator, is identified by searching again the correct capability using the received assigned number. Once found the capability, the code stored in the default measure extension is extracted.

The last block of the algorithm performs an alignment, in the case the extracted unit of measure is different from the received one on the BLE channel (e.g. cm/s to m/s). Indeed, the used sensor can operate using different scale and, sometime, different unit of measure. After the the received value has been adapted, it is added to the observation resource in the value property together with the corresponded unit of measure code.

6 BLE reading module extension mechanism

The approach described above offers a mechanism (1) to link a measurement data (i.e. BLE *value field*) to a FHIR DeviceDefinition *capability* and (2) to state its meaning through the usage of the its *type* attribute (relying on well-known health vocabularies e.g. SNOMED, LOINC). In other words the conceptual mapping and the algorithm implemented by the FHIR Adapter (see Sect. 5), assumes that BLE raw data are always properly acquired and read by the *ProtocolManager* component.

However, the BLE standard leaves several customization possibilities to the devices manufacturers so that it is not possible to guarantee that such component will be able, without any adaptation, to read data coming from new plugged sensors depending from the different telemonitoring contexts. For this reason, we have envisaged a mechanism to extend the default capability of the *ProtocolManager* to support BLE devices not compliant with GATT standard profiles.

In our sensor home gateway, we have integrated a reader which is able to read the GATT profiles (standard or not) while, about not compliant devices, a plug-in management system has been designed to make the application independent from new devices appearing over time in the environment. Usually, in fact, when a new sensor becomes available, the Home Gateway developer should release a new

version making it available on the market (e.g. iOS app store, Google play store), forcing the final user to perform a version update. This traditional approach has some limits.

For instance, the update could not always happen automatically (for example because the automatic update option is disabled) and this would not allow to receive data from the new sensors. Moreover, it requires, in any case, the gateway must be stopped interrupting, consequently, the sensors data acquisition. Last but not least this approach forces the developer (1) to fully understand the new sensors BLE specifications and (2) to adapt, accordingly, the gateway mobile app.

For all these reasons, it is clear that a system that introduces an adequate level of abstraction between the gateway and all the BLE module readers would be the best solution to guarantee high system reliability. In the Fig. 7, all the principal blocks composing the Bluetooth Plugin Manager (BLEPM) ecosystem are reported. The FHIR Server is the container for device definition resource and measurements produced by sensors. The “BLEPM” is the micro-service responsible of all plugin regulation process enabling the correct dialogue between Sensor Home Gateway and its registered sensors. Last two blocks represent the “Sensor Home Gateway” (i.e. the core of our solution) and the group of in home installed sensors. To better understand the devices management process and the benefits of our solution, we will analyze the interactions between the Sensor Home Gateway, the sensors and the BLEPM in two different scenarios:

- *Scenario 1:* The sensor is compliant with standard GATT profiles or it is a custom sensor (i.e. not compliant with standard profiles) but the gateway app was already modified to integrate the custom BLE reading module for that sensor.
- *Scenario 2:* The sensor is custom (i.e. not compliant with standard profiles) and the gateway does not yet integrate a custom BLE reading module for that sensor.

In the first scenario, we can think that the connected sensor is one of the fully compliant sensor to the GATT profiles. In this case, our integrated reader, already included in the gateway, is able to directly read the BLE data. We are in this scenario also if a custom BLE reader has already been integrated in the gateway release currently available on the market. For instance, we are in this case if we wanted to officially support a specific device upon gateway mobile app software delivery.

In the second scenario, the gateway is aware about the DeviceDefinition related to the sensor but is not able to read its data over BLE channel. The gateway needs a tool that enables the reading process and, consequently a custom reader (AKA Plug-In) must be dynamically downloaded/updated.

This “Plug-In” is requested to BLEPM micro-service, that can return to the gateway one of these object:

- An Android dex library containing all code necessary to enable the reading process.
- A GATT xml file containing the BLE characteristic.
- An error message if the sensor is not yet supported.

The BLEPM micro-service is a service for the devices manufacturers, these users will be able to upload anytime their produced plugins. During the upload process it is mandatory to provide the following information: (1) the BLE service uuid, (2) the manufacturer name, (3) the file that will enable the reading process.

The received file will be processed to manage the versioning and to manage the format if it is not an XML file. In fact, an Android application can not run byte-code organized in an aar package, for this reason, the BLEPM service executes a compiling process over the code contained in the received aar package to turn it into a dex library compliant with the Android system.

When the sensor home gateway requests a plugin for a new device, it preliminary queries the BLEPM service that sends details about the available plug-in that represents the BLE service supported by the new device. These details contain info about the plugin file and relative version.

After that, the gateway sends a new request to download the plug-in file. This two step request is useful to avoid any download when the gateway already holds the latest plugin for that sensor. In case a new version becomes available the gateway will proceed with the plugin file download updating its local information about the plugin.

If the received plugin is an xml file the Gateway will be able to use it during the reading process through the integrated reader.

The reading process, in case of dex file, can correctly operate because we have developed and distributed a framework as an Android aar library. Every device manufacturer can import this library in Android Studio and use it to develop its custom plugin. After the development, the manufacturer can upload the exported aar custom library through the BLEPM service. Once uploaded the file will follow the compilation process, already described above, to transform it in a dex library.

7 Conclusion

This paper proposed a FHIR-based multi protocol IoT Home Gateway, consisting in an Android mobile app acting as the collecting node of sensors data received over different protocols and from remote cloud service (i.e. GoogleFit) providing a seamless integration of such data into a FHIR based HIS.

The main aspects of the architecture are the interoperability and the extension simplicity. It allows transparent interoperability in a way that devices can use its own protocol (currently BLE, MQTT) to have a heterogeneous communication. It was conceived to be extensible as well: enabling a new sensors is only a matter to edit a new FHIR DeviceDefinition and, if needed, to upload a BLE reader plugin.

Finally, there are some ongoing work. Although the MQTT reading module has been integrated in the gateway, work is still to be completed to adapt the user interface for supporting the DeviceDefinition editing for such kind of devices. The testbed used till now, composed of an Android tablet/smartphone (Android from V7 to V10) and several devices—the environmental parameters sensor (from ST Microelectronics), the ultrasound sensor (from University of Firenze), XIAOMI Band 4 and 3 of energy consumption, CPU and memory usage. However, a more detailed analysis should be carried out on this aspect and we aim at arriving to a clearer understanding by enrolling a broader range of sensors in a future work.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Mavrogiorgou A, Kiourtis A, Perakis K, Pitsios S, Kyriazis D (2019) Iot in healthcare: Achieving interoperability of high-quality data acquired by IOT medical devices. *Sensors* (Basel, Switzerland) 19(9):1978. <https://doi.org/10.3390/s19091978>. <https://pubmed.ncbi.nlm.nih.gov/31035612> [31035612[pmid] and PMC6539021[pmcid] and s19091978[PII] (Online)]
2. Oniga B, Munteanu A, Dadarlat V (2018) Open-source multi-protocol gateway for internet of things. In: 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet), pp 1–6
3. Aloï G, Caliciuri G, Fortino G, Gravina R, Pace P, Russo W, Savaglio C (2016) A mobile multi-technology gateway to enable iot interoperability. In: IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), pp 259–264
4. Aloï G, Caliciuri G, Fortino G, Gravina R, Pace P, Russo W, Savaglio C (2017) Enabling iot interoperability through opportunistic smartphone-based mobile gateways. *J Netw Comput Appl* 81:74–84. <https://www.sciencedirect.com/science/article/pii/S1084804516302405> (Online)

5. Ahmad RW, Bashir RS, Saeed S, Lee Y, Ko K, Son Y (2017) Online cloud-based battery lifetime estimation framework for smartphone devices. *Procedia Comput Sci* 110:70–77. <https://doi.org/10.1016/j.procs.2017.06.118> (Online)
6. da Cruz MA, Rodrigues JJ, Lorenz P, Solic P, Al-Muhtadi J, Albuquerque VHC (2019) A proposal for bridging application layer protocols to http on iot solutions. *Future Gener Comput Syst* 97:145–152. <https://doi.org/10.1016/j.future.2019.02.009> (Online)
7. Sciancalepore S, Piro G, Vogli E, Boggia G, Grieco L, Cavone G (2016) Licitus: A lightweight and standard compatible framework for securing layer-2 communications in the IOT. *Comput Netw* 108:66–77. <https://doi.org/10.1016/j.comnet.2016.08.003> (Online)
8. Kumar P, Dezfouli B (2019) Implementation and analysis of quic for mqtt. *Comput Netw* 150:28–45. <https://doi.org/10.1016/j.comnet.2018.12.012> (Online)
9. Langley A, Riddoch A, Wilk A, Vicente A, Krasic C, Zhang D, Yang F, Kouranov F, Swett F, Iyengar J et al (2017) The quic transport protocol. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM. <https://doi.org/10.1145/3098822.3098842> (Online)
10. Sloane EB, Cooper T, Silva R (2021) Mdira: IEEE, ihe, and fhir clinical device and information technology interoperability standards, bridging home to hospital to “hospital-in-home. *South-eastCon* 2021:1–4
11. Decree n.267 gazzetta ufficiale (2018) Specialization area: Technologies for life environments, code ars01_00900, november 2018
12. Naeem M, Paragliola G, Coronato A (2021) A reinforcement learning and deep learning based intelligent system for the support of impaired patients in home treatment. *Expert Syst Appl* 168:114285. <https://www.sciencedirect.com/science/article/pii/S095741742030988X> (Online)
13. Coronato A, Cuzzocrea A (2020) An innovative risk assessment methodology for medical information systems. *IEEE Trans Knowl Data Eng.* <https://doi.org/10.1109/TKDE.2020.3023553>
14. Paragliola G, Coronato A (2018) Gait anomaly detection of subjects with Parkinson’s disease using a deep time series-based approach. *IEEE Access* 6:73280–73292
15. Paragliola G, Coronato A (2021) An hybrid ECG-based deep network for the early identification of high-risk to major cardiovascular events for hypertension patients. *J Biomed Inf* 113:103648
16. Google fit health data types (2021). <https://developers.google.com/fit/datatypes/health> (Online)
17. Samsung health data type (2021). <https://developer.samsung.com/health/android/data/guide/health-data-type.html> (Online)
18. Samsung health sdk for device (2021). <https://developer.samsung.com/health/device/overview.html> (Online)
19. Samsung health device specs (2019). <https://developer.samsung.com/health/device/device-specs.html> (Online)
20. OAuth api verification faqs (2020). <https://support.google.com/cloud/answer/9110914> (Online)
21. Google fit developer and user data policy (2021). <https://developers.google.com/fit/policy> (Online)
22. Android vs. ios (2021). <https://www.kantarworldpanel.com/smartphone-os-market-share/> (Online)
23. The unified code for units of measure (2021). <https://ucum.org/trac> (Online)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.